Encapsulation Application Research of ArcSDE Access Interface in .Net Environment

Min FENG, Qingsheng SHANG, Jianwen GUO, Yingchun GE
Cold and Arid Regions Environmental and Engineering Research Institute
Chinese Academy of Science
Lanzhou, China
diverge@163.com

Abstract—**The GIS and Digital Roadbed System of the Qinghai-Tibet Railroad is developed in the .Net environment and designed to store and to manage the information along the Qinghai-Tibet railroad, which is 1118km long and is constructed in the regions more than 4000m high. Spatial data Management plays a very important role in this GIS, and ArcSDE was selected to implement the management of mass spatial data. How to connect ArcSDE and exchange vector and raster spatial data in this system is a key technique. There are three approaches, which are recommended officially, can be used for client to connect ArcSDE, but none of them can meet the needs of spatial information storage and management in the GIS and Digital Roadbed System of the Qinghai-Tibet Railroad. This paper explained a solution of encapsulating the ArcSDE Client API for C programmers into the form of .Net managed code with C++ programming language. The final program not only provides a solution to resolve the problem of spatial data management, but also bring forward a flexible interface which followed the principle of OOP.**

Keywords- Spatial Database; System Encapsulation; .Net; Qinghai-Tibet Railroad

I. INTRODUCTION

A. Background

Spatial database is very important for every Geographic Information System (GIS), and it is the key difference between GIS and general information system [1]. Spatial database can be classified into two kinds: local spatial database and distributed spatial database. The local one is appropriate to the situations that the system do not need to access data through network and is unsuitable to deal with the mass data for the limitation of client computer. However, distributed spatial database is much flexible: you can separate the spatial database from the GIS client and put it on another computer. So you can offer the spatial database with high performance computer, in order to give full play to the spatial database. Distributed spatial database also share data between different clients, and those clients don't need store the duplicate of the data, in addition, the modification of the spatial data can be reflected to all clients.

ArcSDE is a comparatively ripe distributed spatial database management system which is developed by ESRI, and it is used to manage the geographical information in the Qinghai-Tibet railroad construction.

B. Ways of accessing data from ArcSDE

Programmers can access data from ArcSDE via three different ways [2]:

- *ArcObjects:* It is a powerful GIS software components library on which ArcMap and ArcCatalog are built. Programmers can access data in ArcSDE through the GeoDatabase data access objects, which is a subset of ArcObjects.
- *ArcSDE Application Programming Interface (API):* ESRI provides two kinds of ArcSDE API: C and Java. These APIs are compliant with the Open GIS Consortium (OGC) simple features specification [2].
- *Structured Query Language (SQL):* While ArcSDE is a gateway between GIS client and relational database, all features in a GeoDatabase are implemented as a set of relational table. Some of these tables represent collections of features. Other tables represent relationships between features, validation rules and attributed domains. So Programmers can access data in those relational tables by SQL directly.

C. Comparison between ways of accessing data

It can give full play to ArcSDE through ArcObjects accessing ArcSDE. Furthermore, ArcObjects can be used in both Win32 and .Net environments, and presents object-oriented interfaces. However, ArcObjects 8.x can not be used with ArcGIS independently, and it is much expensive; in addition, it needs need many computer resources to run it. So it does not suit the requirements of system developing using ArcObjects singly to access spatial data; Accessing GIS data through SQL is much inconvenient, because you have to understand how the GIS data is stored in relational database and it is dangerous to destroy the GeoDatabase if you make some mistakes when you try to modify some GIS data or relationships. Finally, we decide to adopt C Client API to access ArcSDE, in order to reduce the development and operation cost.

Another question we meet is that GIS and Digital Roadbed System of the Qinghai-Tibet Railroad is developed in .Net Environment, but ESRI does not provide ArcSDE API for .Net. Although we can import the C API to C# source file, it is too difficult because there are about 30 C structures, 21 enumerations and 757 functions [3]. We can not ensure all those elements will work well after being imported. Unlike

This work is supported by the innovation project of CAS (KZCX1-SW-04), innovation project of CAREERI, CAS (220014-016A)

Report Documentation Page		*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **25 JUL 2005**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **Encapsulation Application Research of ArcSDE Access Interface in .Net Environment**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Cold and Arid Regions Environmental and Engineering Research Institute Chinese Academy of Science Lanzhou, China**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001850, 2005 IEEE International Geoscience and Remote Sensing Symposium Proceedings (25th) (IGARSS 2005) Held in Seoul, Korea on 25-29 July 2005.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**4**	

Win32 environment, all code written for .Net program will be compiled to Common Intermediate Language (CIL), which finally runs on the Common Language Runtime (CLR). .Net programs do not locate or release memory directly; CLR does this work [4]. If we import C library into .Net Environment, we must transfer information between Win32 and .Net Environments, and it is not an easy task.

C++ is derived from C programming language, and it is compatible with C. Microsoft developed Visual Studio .Net in 2002, and extended C++ to support .Net programming, which is called Managed C++ [4]. We think Managed C++ is the best choice to resolve the problem of integrating ArcSDE C API with the GIS and Digital Roadbed System of the Qinghai-Tibet Railroad.

II. Objects

We try to encapsulate ArcSDE C API into .Net Assembly, which can be loaded to .Net Environment, and meet the needs of managing GIS data in GIS and Digital Roadbed System of the Qinghai-Tibet Railroad. The work of encapsulation does not only transfer the functions of ArcSDE C API to the object system, but also design an OOP (Object-Oriented Programming) profile of C API architecture.

III. Architecture of ArcSDE C API

The architecture of ArcSDE C API follows the Simple Feature Specification of OGC [2]. All ArcSDE C API Functions can be divided into 12 groups, and each group includes many functions, structures and enumerations [5].

A. Groups of ArcSDE C API

- *Database Connections and Server Instances:* build connection between client and ArcSDE, on which almost all operations are built.
- *Vector Layer Management:* Layer is a fundamental element for GIS data management; this group of ArcSDE C API offers operations on layers, such as adding, deleting layer and modifying layer's attributes.
- *Geometry Function:* offer operations on geometry, such as creating, decompounding features and retrieving the topologic relationship between features.
- *Coordinate Reference:* offer operations to define coordinate reference and spatial reference, and modify their attributes.
- *Projection Engine:* offer operations to define projection, as well as functions to project GIS data between two different projections.
- *Stream and Query Management:* Stream is used to transfer data between ArcSDE and client, so it is very fundamental for implementing queries and modifies on GIS data.
- *Table Management:* provide functions to manage the non-spatial tables.
- *Log files:* offer functions on log information.
- *Raster Data Management:* offer operations on GIS data which stored in GIS raster form.
- *Versioned Database Management:* offer operations on ArcSDE versions, which are used to provide more flexible mechanism on modifying GIS data.
- *Locators:* offer supports on address geocoding.
- *Application Development Support:* offer supports on program debug.

IV. Schemas of Encapsulating ArcSDE C API into .Net Environment

Two schemata were designed to encapsulate ArcSDE C API into .Net environment. "Fig 1" shows those two schemata.

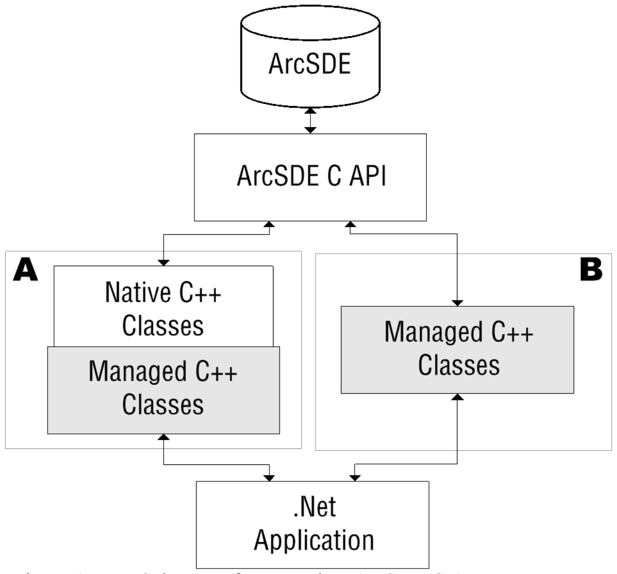

Figure 1. Two Schemas of Encapsulate ArcSDE C API to .Net

A. 2-tier schema

There are 2 tiers of C++ codes between ArcSDE C API and .Net application. One is written by native C++, and provokes the ArcSDE C API directly; the other is written by managed C++, and all classes in the latter are located under the SDE_Client namespace. Each class of this tier comprises the corresponding class of the former tier, and takes charge to transfer information between Win32 environment and .Net CRL. Each operation provoked by .Net clients is processed by managed C++ classes which convert the managed information to win32 format, and then is passed to the native classes which call the ArcSDE C API. After the operation finished the result transfers to the .Net clients reversely.

B. 1-tier schema

Only one tier is designed in this schema, and the classes in the tier not only transfer managed information between Win32 environment and .Net CRL, but also pass the parameters to ArcSDE C API and bring the result back to the .Net Client.

V. COMP ARISON BETWEEN TWO SCHEMATA

According to the comparison, we finally decided to adopt 1-tier schema to implement the encapsulation of raster management, in respect that raster has more complex Relationships between classes, and the performance is much important; and adopt 2-tier schema to implement the encapsulation of others, for it fits both Win32 and .Net environments.

TABLE 1. COMPARISON BETWEEN TWO SCHEMATA

Schema	Advantages	Disadvantages
A	◆ If the managed C++ tier is removed, the native C++ tier can be used in Win32 Environment; ◆ Each layer does its own job, so the errors are easy to handle.	◆ Every operation must be implemented in two tiers, so it is inconvenient to handle the complex Relationships between the classes; ◆ The workload is relatively large;
B	◆ It can implement easily complex Relationships between classes; ◆ There is less workload of the 1-tier schema than 2-tier schema; ◆ Performance is much better.	◆ It can't be used in Win32 environment; ◆ It is apt to make mistakes.

VI. ARCHITECTURE OF ENCAPSULATION

A. Modules

All components can be classified into 7 groups after being encapsulated:

- *Connection*: Encapsulates the ArcSDE C API related to connections, maintaining the connections between clients and ArcSDE, establishing concurrency control for threaded client applications, and managing transactions.
- *Stream*: Encapsulates the ArcSDE C API related to stream, and providing functions to query, add, delete and modify the features and tables. It also supports different data types, such as blob, double, date, float, integer, string, smallint, as well as shape.
- *Table*: Encapsulates the ArcSDE C API related to non-spatial table, and providing functions to create, modify and delete tables and structures.
- *Shape*: Encapsulates the ArcSDE C API related to geometry and providing functions to generate and decompose shapes of features
- *Projection*: Encapsulates the ArcSDE C API related to coordination reference and projection to support operations about projection.
- *Layer*: Encapsulates the ArcSDE C API related to GIS layer.
- *Raster*: Encapsulates the ArcSDE C API related to Raster data management. ArcSDE manages GIS Raster data by the system which is consisted of Column, Raster, Band and Tile, so we designed a series of classes to represent it.

B. Relationships between modules

"Fig 2" shows the relationships between classes and sub systems, and each sub system is a collection of several classes which bear on one subject.

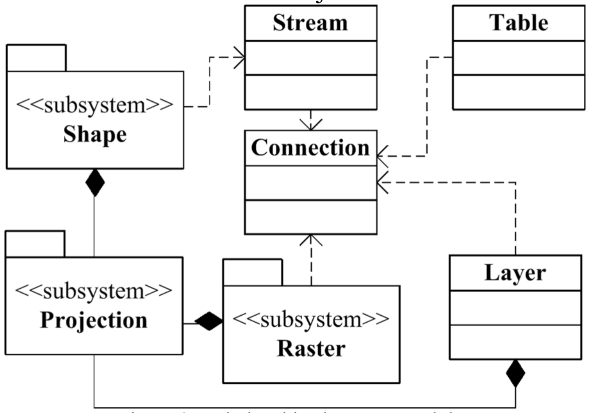

Figure 2. Relationships between modules

C. Key points of design

1) Inheriingt from System::IDisposable Interface

.Net CRL manages memory for .Net application and .Net Garbage Collection (GC) will find and release all memory at some time [8], so all our classes inherit from System::IDisposable interface, which defines a single function:

```
Void Dispose();
```

The function is designed to be called when the resource in the class should be released; furthermore, C# provides another way to call the Dispose function:

```
Layer _layer = new Layer();
using(_layer)
{
    // Scope of action
    ...
}
//Dispose will be called automatically by CRL
```

2) Encapsulating ArcSDE C API

Each ArcSDE C API function does a single job, but some jobs can be put together, especially a pair of operations on one item, among which one is to retrieve value from that information and the other put value to it. Attribute is a kind of method which is supported by some programming languages such as C#, Java etc. Although each Attribute consists of two functions: "get" and "put", they together act as a single variable, so we encapsulate each pair of functions on one item to a single Attribute. There is an example below:

```
#pragma managed
namespace SDE_Client{
        __gc public class Layer:public System::IDisposable{
        public:
                // Get and set the array size of layer
                __property long get_ArraySize(){
                        return this->m_Handle->GetArraySize();
                }
                __property void set_ArraySize(long optimalArraySize){
                        return this->m_Handle->SetArraySize(optimalArraySize);
                }
                //...
        };
}
```

3) Class Initialization

Some ArcSDE C API functions are very special because they just make preparations for following functions or release the resources used by foregoing functions, so they ought to be encapsulated to the constructor of destructor function. However, native C++ requires there should not make exceptions in constructor or destructor functions [9], and those functions do not suit to be called in native C++ constructor or destructor function. So a Create function is provided to assign resources and a Close function is provided to release resources. But managed C++ does not have this limitation, for it can release unused memory automatically, and we move the Create function to the constructor function of class, and Close function to the destructor function. Accordingly, the classes of managed codes are really user – friendly.

4) Handle Exception

All error information are brought back by the return value after ArcSDE C API is called, so we have to analyze the value to understand what is wrong with it and it need many codes to analyze it. Some programming languages support exceptions which are much flexible and can bring back more information about the error than just a single value. We designed SEException class. When it meets an error, an instance of that class will be created and sent to the "catch" codes which can accept the exception type. The class has several attributes, and each of them store a kind of information about the error.

```
class SEException
{
public:
        ...
        //Error code
        Long m_Rc;
        //Comment of the error
        std::string     m_Comment;
        //Extended error code
        Long m_ExtError;
        //Description of the error
        std::string     m_ErrorString1;
        //Extended description of the error
        std::string     m_ErrorString2;
        ...
};
```

5) Structures and Enumerations

There are lots of C structures and enumerations defined in the ArcSDE C API library, each of them represent a kind of information. In the work of encapsulation, some structures were mapped into C++ classes, such as Stream, Connection etc, and others are yet defined as C++ structures and enumerations, for example, the GIS point:

```
public struct Point
{
        double x;
        double y;
        double z
};
```

6) Auxiliary Functions

Some auxiliary functions are very useful to the work of encapsulation, and those functions are developed to offer supports of converting string from Win32 to .Net environment; return value to Exception and so on.

VII. SUMMARY

Encapsulating ArcSDE C API by managed C++ can be used not only in .Net environment, but also in Win32 environment by a few modifications. It meets the need of GIS data managements of GIS and Digital Roadbed System of the Qinghai-Tibet Railroad, and provides an available choice for the future GIS development which also uses ArcSDE to manage GIS data. The Encapsulation is more than conversion from C to C++, but a design of the architecture following OOP.

ACKNOWLEDGMENT

This work is supported by the innovation project of CAS (KZCX1-SW-04), innovation project of the Cold and Arid Regions Environmental and Engineering Research Institute, CAS (220014-016A).

REFERENCE

[1] Wang J-Y. Principles of Spatial Information System [M]. Beijing: Science Press, 2001

[2] Zeiler M. Modeling Our World [M]. Environmental Systems Research Institute, Inc. 1999.

[3] Environmental Systems Research Institute Inc. ArcSDE Developer Help [R]. Environmental Systems Research Institute, Inc. 2002.

[4] Microsoft Corporation. MSDN Library for Visual Studio .NET 2003 [R]. Microsoft Corporation. 2003.

[5] West R. Understanding ArcSDE [M]. Environmental Systems Research Institute, Inc. 2002.

[6] Harris M. Managing ArcSDE Services [M]. Environmental Systems Research Institute, Inc. 2002.

[7] Open GIS Consortium, Inc. Simple Features Specification for SQL Revision 1.1 [S]. Open GIS Consortium, Inc. 1999.

[8] Thai T. .NET Framework Essentials [M]. O'Reilly, 2001.

[9] Eckel B. Thinking in C++ [M]. Prentice Hall, 1995.